Национальный исследовательский университет «МИЭТ»

Отчет о выполненной лабораторной работе №1

По предмету: Технология программирования MPI

На тему:Передача и прием сообщений в MPI

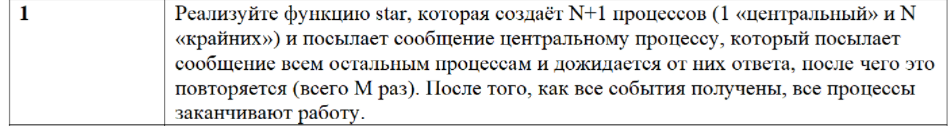
Цель: изучить основные принципы приема и передачи сообщений в технологии MPI на примере использования в рамках языка С++.

Выполнила Марина Алина

Группа ПИН-34

Вариант №1

01.03.2022



//------------Марина Алина ПИН-34 1 вариант------------------------

//Реализуйте функцию star, которая создаёт N+1 процессов (1 «центральный» и N «крайних»)

//и посылает сообщение центральному процессу, который посылает сообщение всем остальным процессам

//и дожидается от них ответа, после чего это повторяется (всего M раз). После того, как все события получены,

//все процессы заканчивают работу.

#include <stdio.h>

#include <mpi.h>

#include "windows.h"

#include <iostream>

#include <iomanip>

#include <string>

#define M 4

using namespace std;

int main(int\* argc, char\*\* argv)

{

int proc\_quantity, rank;

MPI\_Init(argc, &argv);

MPI\_Status status;

// Получаем количество процессов и их текущий ранг

MPI\_Comm\_size(MPI\_COMM\_WORLD, &proc\_quantity);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

srand(time(NULL) + rank);

//как много процессов используется

if (rank == 0)

cout << "Num of all processes: " << proc\_quantity << endl;

//Proccesses cycle

for (int loop = 0; loop < M; loop++)

{

int random\_value = rand() % 100;

if (rank == 0)

{

printf("%d: sent out %d (loop %d)\n", rank, random\_value, loop);

for (int i = 0; i < proc\_quantity - 1; i++)

// Послать сообщение всем крайним процессам

MPI\_Send(&random\_value, 1, MPI\_INT, i + 1, 0, MPI\_COMM\_WORLD);

for (int i = 0; i < proc\_quantity - 1; i++)

// Принять сообщения от всех крайних процессов

MPI\_Recv(&random\_value, 1, MPI\_INT, MPI\_ANY\_SOURCE, 0 /\*MPI\_ANY\_TAG\*/, MPI\_COMM\_WORLD, &status);

}

else

{

// Принять сообщение от главного процесса

MPI\_Recv(&random\_value, 1, MPI\_INT, 0, 0, MPI\_COMM\_WORLD, &status);

printf("%d: received %d (loop %d)\n", rank, random\_value, loop);

int confirmation\_value = random\_value + rank;

//Послать сообщение главному процессу

MPI\_Send(&confirmation\_value, 1, MPI\_INT, 0, 0, MPI\_COMM\_WORLD);

//printf("%d: sent out %d (loop %d)\n", rank, confirmation\_value, loop);

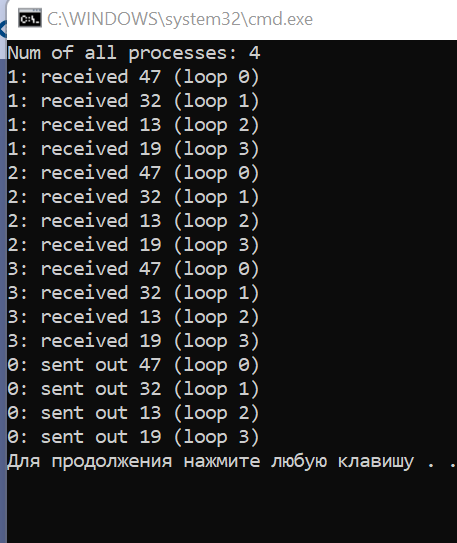
}

}

MPI\_Finalize();

return 0;

}



**Контрольные вопросы**

1.В чем состоят основы технологии MPI?

Основу MPI составляют операции передачи сообщений. Среди предусмотренных в составе MPI функций различаются **парные** (**point-to-point**) операции между двумя процессами и **коллективные** (**collective**) коммуникационные действия для одновременного взаимодействия нескольких процессов.

2.В чем состоят основные преимущества и недостатки технологии MPI?

**Преимущества:**  
- MPI помогает решить проблему переносимости параллельных программ между разными компьютерными системами.  
- MPI содействует повышению эффективности параллельных вычислений: практически для каждого типа вычислительных систем существуют реализации библиотек MPI, учитывающие возможности используемого коммуникационного оборудования.  
- MPI облегчает процесс написания параллельных программ, когда при разработке используются библиотеки программных модулей, написанных с использованием этого интерфейса.  
- MPI позволяет создавать хорошо масштабируемые параллельные программы.  
  
**Недостатки:**  
- MPI является низкоуровневым инструментом программиста.  
- Не существует реализаций MPI, в полной мере обеспечивающих совмещение обменов с вычислениями.  
- MPI не предоставляет механизмов задания начального размещения процессов по процессорам.  
- Полномасштабная отладка MPI-программ затруднительна вследствие одновременного исполнения нескольких программных ветвей.

3.Что понимается под параллельной программой в рамках технологии MPI?

Под **параллельной программой** в рамках MPI понимается множество одновременно выполняемых **процессов**. Процессы могут выполняться как на разных процессорах, так и на одном. Каждый процесс параллельной программы порождается на основе копии одного и того же программного кода (**модель SPMD).** Все процессы программы последовательно перенумерованы от 0 до **p-1**, где **p** есть общее количество процессов. Номер процесса именуется **рангом** процесса.

4.Как происходит инициализация и завершение MPI программ?

**Первой вызываемой функцией** MPI должна быть функция:

int MPI\_Init(int \*agrc, char \*\*\*argv);

для инициализации среды выполнения MPI-программы. Параметрами функции являются количество аргументов в командной строке и текст самой командной строки.

**Последней вызываемой функцией** MPI обязательно должна являться функция:

int MPI\_Finalize(void);

5.Как происходит передача и прием сообщений MPI программе?

Для **передачи сообщения** процесс-отправитель должен выполнить функцию:

int MPI\_Send(void \*buf, int count, MPI\_Datatype type, int dest,

int tag, MPI\_Comm comm);

Для **приема сообщения** процесс-получатель должен выполнить функцию:

int MPI\_Recv(void \*buf, int count, MPI\_Datatype type, int source,

int tag, MPI\_Comm comm, MPI\_Status \*status);